TAMPEREEN
AMMATTIKORKEAKOULU

WORDPRESS-TEEMAN KEHITYS PIENEN IT-TALON KÄYTTÖTARPEISIIN

Tomi Mäkinen

Opinnäytetyö
Joulukuu 2017
Tietojenkäsittely
Digitaalinen media

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittely
Digitaalinen media

MÄKINEN, TOMI
WordPress-teeman kehitys pienen IT-talon käyttötarpeisiin

Opinnäytetyö 39 sivua
Joulukuu 2017

Tämän opinnäytetyön tavoitteena oli parantaa WordPress-sisällönhallintajärjestelmällä toteutettujen verkkosivujen kustannustehokkuutta pienten IT-yritysten tuotannossa. Opinnäytetyönä kehitettiin toimeksiantajan käyttöön oma ulkoasuteema, jonka päälle WordPress-sivustot voidaan toteuttaa.

WordPress on kirjoitushetkellä maailman suosituin sisällönhallintajärjestelmä nettisivujen kehityksessä. WordPress perustuu avoimeen lähdekoodiin ja on ilmainen ja siten kustannustehokas ratkaisu verkkosivujen luomiseen. WordPress-sivustoja pystyy päivittämään ilman ohjelmointitaitoja, mikä lisää niiden käyttäjäystävällisyyttä.

Teema toimii pohjana sivustojen toteutuksessa. Teema määrittää sivustojen rakenteen ja lähtökohdan ulkoasun toteutukselle. Valmiita teemoja on saatavilla WordPressin ilmaisjakelusta ja maksullisina kolmannen osapuolen verkkokaupoista. Kummassakin on omat haittapuolensa, jotka laskevat verkkosivuprojektien kustannustehokkuutta.

Raportointi selvittää teeman kehityksen kannalta tärkeät vaiheet ja WordPressin ominaisuudet, sekä esittelee mahdollisia jatkokehitysajatuksia. Lisäksi työ käsittelee muita hyödyllisiä ominaisuuksia, jotka parantavat teeman käytettävyyttä. Opinnäytetyön pohjalta on mahdollista toteuttaa oma WordPress-teema, jota voi käyttää ainakin lähtökohtana verkkosivujen toteutukseen työelämässä tai vapaa-ajalla.

Asiasanat: WordPress, teema, verkkosivut, verkkosuunnittelu

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Digital Media

MÄKINEN, TOMI
Developing a WordPress Theme for the Use of an IT Company

Bachelor's thesis 39 pages
December 2017

The purpose of this thesis is to increase the cost-effectiveness of designing websites for the clients of small to medium IT companies by developing a WordPress theme. A theme is used as a basis to create the design and functionality of all WordPress sites.

At the time of writing this thesis, WordPress is the most popular content management system used for website development. Because it is free to use and based on open-source code, it is a good choice for developing client websites. Site content management and updating can be performed without any programming skills.

Free and commercial themes developed by third parties have their own disadvantages in regards to the time it takes to find suitable ones for each project and learning how to modify them and, in the case of commercial themes, the financial cost of licensing them. A self-made theme lets you create sites upon a free and familiar foundation.

This thesis analyzes the most important phases of theme development and WordPress' functions and properties. It also includes thoughts on potential features for future development. These are likely to be useful for those familiar with or looking to pick up WordPress for their business or creating personal websites.

Key words: WordPress, theme, web development, websites

SISÄLLYS

LYHENTEET JA TERMIT

CSS	Cascading Style Sheets, eli suomeksi porrastetut tyyliarkit, joita käytetään määrittämään sivustojen ulkoasua. Samoja elementtejä käsittelevät tyylisäännökset yhdistetään niin, että ne, joissa on tarkemmin määritelty valitsin, ohittavat laajemmin vaikuttavat tyylisäännökset.
JavaScript	Alun perin selaimessa suoritettava ohjelmointikieli, jolla tehdään sivustoille dynaamista, käyttäjän valintoihin perustuvaa toiminnallisuutta. Nykyisin myös mahdollista käyttää palvelinpuolen ohjelmoinnissa.
MySQL	WordPressin käyttämä tietokantaohjelmisto, jolla sivuston sisältö, käyttäjät ja asetukset tallennetaan muistiin.
ohjausnäkymä	WordPressin back end –hallintapuoli, josta muun muassa päivitetään sivustoa ja muokataan sen asetuksia.
PHP	Palvelinpuolen ohjelmointikieli, jolla sivustot ladataan dynaamisesti esimerkiksi hakemalla sivun sisältö palvelimen tietokannasta.
polkutunnus	URL-muotoinen tunnus, jolla viitataan yksittäiseen sivuun tai muuhun sisältötyyppiin.
vimpain	WordPressin valmis sisältöelementti, jota pystytään muokkaamaan ja jonka voi sijoittaa ilman koodituntemusta sivuston teemassa määritellyille alueille ohjausnäkymästä. Esimerkkejä: sivupalkin valikko, kirjautumislomake, lista viimeisimmistä artikkeleista.

1 JOHDANTO

Tämä opinnäytetyö käsittelee oman teeman kehittämistä WordPress-sisällönhallintajärjestelmälle. Teema määrittelee lähtökohdat sivuston julkisen puolen rakenteen, ulkoasun ja toiminnallisuuden toteuttamiseen ja on täten kriittinen osa WordPress-sivustoprojekteja. Opinnäytetyössä tutustutaan teeman perusrakenteeseen, teeman ohjelmointiperiaatteisiin ja teeman kehityksen kannalta oleellisiin asioihin WordPressin toiminnassa.

Valmiita WordPress-teemoja löytyy sekä maksullisina että ilmaisina versioina lukematon määrä. Kolmannen osapuolen teemojen käytössä on kuitenkin omat ongelmansa: maksulliset teemat alentavat sivustoprojektien kustannustehokkuutta ja ilmaiset teemat ovat yleensä rajallisia ominaisuuksiltaan ja rakennettu hyvin tiukasti yhden ulkoasuratkaisun ympärille. Niin ilmaisen kuin maksullisenkin teeman versiopäivitykset ja tuki saattavat loppua yllättäen, mikä voi johtaa yhteensopivuusongelmiin tulevien WordPress-versioiden ja lisäosien kanssa. Lisäksi omien laajennuksien ja päivitysten tekeminen kolmannen osapuolen teemaan vaatii tutustumisen alkuperäiseen koodiin, joka pahimmillaan on tahallisesti tehty vaikeaselkoiseksi.

Oman teeman kehitys tarjoaa ratkaisun näihin ongelmiin ja myös muita etuja. Tuttu teemapohja helpottaa sivustokohtaisten muutoksien tekemistä ja vianmääritystä ongelmatilanteissa. Teeman kehitystä voi jatkaa sivustoprojektien mukana rakentamalla teemaa modulaarisista osista, joita voi yhdistellä eri sivuprojektien tarpeiden mukaan. Samalla kartutetaan myös omaa WordPress- ja ohjelmointitietämystä. Tutustuminen WordPressin toimintaperiaatteisiin helpottaa myös lisäosien ja uusien ulkoasuratkaisujen kehitystä.

2 TAUSTA

2.1 WordPress kehitysalustana

Vuonna 2003 ensikertaa julkaistu, avoimeen lähdekoodiin perustuva WordPress tunnettiin aluksi lähinnä blogialustana. Sisällönhallintajärjestelmän suuntaan WordPress kehittyi vuonna 2005 julkaistujen 1.5 ja 2.0 versiopäivitysten myötä, jolloin esimerkiksi staattiset sivut tulivat käyttöön. WordPress on tällä hetkellä internetin suosituin sisällönhallintajärjestelmä. Noin neljännes maailman julkaistuista verkkosivuista on toteutettu WordPressillä (Built With n.d.; W3 Techs n.d.).

WordPressin etuina ovat ilmaisuuden lisäksi helppokäyttöinen sisällönhallinta, jolla sivustoa voi päivittää graafisessa ohjausnäkymässä ilman ohjelmointitietämystä. WordPressiin on saatavilla lukematon määrä lisäosia laajentamaan sen toiminnallisuutta. Ilmaisten lisäosien kehittäjät voivat ladata lisäosansa Wordpress.orgin säilöön, josta sivuston kehittäjät voivat asentaa ja aktivoida ne suoraan sivuston ohjausnäkymästä. Lisäosilla WordPress-sivustolle on mahdollista toteuttaa ilman koodikieltä esimerkiksi verkkokauppa, keskustelufoorumi tai intranet. WordPress tarjoaa myös hyvät lähtökohdat hakukoneoptimoitujen sivustojen toteutukseen (Lyngbø 2015).

WordPressin minimivaatimukset palvelinympäristöltä ovat PHP:n versio 5.2.4 ja MySQL:n versio 5.0. Uusimpien versioiden käyttö kuitenkin on suositeltua. WordPress on aktiivisessa kehityksessä, kirjoitushetkellä viimeisin versiopäivitys julkaistiin 19.9.2017.

2.2 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi Trival Oy, mainos- ja mediatalo, joka päätoimisesti toteuttaa asiakkailleen verkkosivuja WordPress-alustalle. Sivustoprojektien toteutusta nopeuttaa valmis teemapohja, jonka päälle sivuston ulkoasua voi kehittää. Opinnäytetyötä varten kehitettiin yrityksen käyttöön ja jatkokehitykseen teema, jonka perusedellytyksenä on responsiivisuus. Teemalla toteutettujen sivujen täytyy mukautua näyttölaitteen leveyden mukaan siten, että ne säilyvät helppolukuisena ja käyttäjäystävällisenä myös mobiililaitteilla. Responsiivisuutta varten teemaan integroidaan ilmainen ja avoimen lähdekoodin front-end-ohjelmistokehys nimeltä Bootstrap.

Teema kehitettiin puhtaalle WordPress-asennukselle Trivalin virtuaalipalvelinympäristöön Plesk-palvelinalustalle, jonka tiedostonhallintajärjestelmää käytettiin teeman toiminnallisuuden ohjelmointiin WordPressin oman muokkaimen lisäksi. Teeman valmistuminen ajoittui huonoon ajankohtaan, sillä sopivaa asiakasprojektia teeman käyttämiseen ei ollut työn raportointihetkellä käynnissä.

3 TEEMAN RAKENNE

3.1 Tiedostorakenne

WordPress-sivustolle asennettavat teemat toteutetaan tai asennetaan omiin alikansioihinsa themes-kansion alle. WordPress tunnistaa themes-kansion sisällä olevat alikansiot, jotka täyttävät teeman kriteerit ja listaa ne sivuston ohjausnäkymässä Ulkoasu-valikon alla. Teemakansiossa olevia PHP- ja CSS-tiedostoja pystyy muokkaamaan WordPressin sisäisellä muokkaimella, jos käyttäjällä on pääkäyttäjän oikeudet.

Teeman toiminnan kannalta vähimmäisvaatimuksena on kaksi tiedostoa, index.php ja style.css. Oletussivupohjana toimiva index.php muodostaa sivun rakenteen ja näyttää sen sisällön siinä tapauksessa, että muita sivupohjatiedostoja ei ole tai ne eivät vastaa sivun sisältötyyppiä. style.css-tyylitiedosto on teeman tärkein tyylitiedosto. Ilman tyylitiedostoa WordPress ei tunnista teemaa ja ilman sivupohjia sivuston julkinen puoli näkyy tyhjänä. Tyylitiedoston ja sivupohjien lisäksi teema useimmiten sisältää functions.php-tiedoston, jossa sijaitsevat teeman käyttämät yleisluokkaiset PHP-funktiot ja mahdolliset liitokset muihin PHP-tiedostoihin.

Mahdolliset JavaScript-tiedostot, lisäfunktiotiedostot, kuvat, kuvakkeet, fontit ja toissijaiset CSS-tiedostot on hyvä lisätä omiin alikansioihinsa selkeyttämään teeman tiedostorakennetta. Kuvassa 1 on esimerkkinä WordPressin vuoden 2017 oletusteeman, Twenty Seventeenin, tiedostorakenne (WordPress n.d.a). Teeman juurikansio sisältää pääsivupohjatiedostot ja oleelliset tiedostot, mutta muut teeman osat on jaettu omiin kansioihinsa. inc-kansio sisältää funktio-tiedostot.

```
assets (dir)
      - css (dir)
      - images (dir)
      - js (dir)
inc (dir)
template-parts (dir)
      - footer (dir)
      - header (dir)
      - navigation (dir)
      - page (dir)
      - post (dir)
404.php
archive.php
comments.php
footer.php
front-page.php
functions.php
header.php
index.php
page.php
README.txt
rtl.css
screenshot.png
search.php
searchform.php
sidebar.php
single.php
style.css
```

KUVA 1: Twenty Seventeen -oletusteeman tiedostorakenne.

Tyylitiedosto style.css sisältää myös teeman tunnistetiedot kommenttikentässä, jossa määritellään teeman tiedot ja käyttöoikeudet. Tärkeimpiä kohtia ovat teeman ja sen tekijän nimi, teeman kuvaus, versio ja tekstitunnus (text domain) lokalisaatiota varten. Teeman käyttämä lisenssi ja osoite, jossa sen voi lukea, ovat myös tärkeitä, jos teemaa on tarkoitus jakaa julkisesti tai myydä. Jotta teeman voi lisätä vapaaseen jakoon WordPress.orgin säilöön, josta sen voi ladata suoraan WordPress-sivustoille ohjausnäkymän kautta, täytyy sen noudattaa vapaan ohjelmiston GNU General Public -lisenssiä. Muita tunnistetietoja ovat teeman ja teeman kehittäjien osoitteet ja yhteistiedot ja avainsanat, joilla teema luokitellaan WordPress.orgin teemasäilön haussa. Lisenssitiedot ja avainsanat (tags) eivät ole tarpeellisia, jos teemaa ei ole tarkoitettu julkiseen jakoon tai myyntiin. Kuvassa 2 on esimerkki vuoden 2017 oletusteeman, Twenty Seventeen, tunnistetiedoista.

```
/*
Theme Name: Twenty Seventeen
Theme URI: https://wordpress.org/themes/twentyseventeen/
Author: the WordPress team
Author URI: https://wordpress.org/
Description: Twenty Seventeen brings your site to life with immersive featured images and subtle animations.
Version: 1.0
License: GNU General Public License v2 or later
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Text Domain: twentyseventeen
Tags: one-column, two-columns, right-sidebar, flexible-header, accessibility-ready, custom-colors, custom-hea
This theme, like WordPress, is licensed under the GPL.
Use it to make something cool, have fun, and share what you've learned with others.
*/
```

KUVA 2: Esimerkki teeman tunnistetiedoista.

Valinnaisesti teemakansio voi sisältää myös screenshot-nimisen PNG-kuvatiedoston, jota käytetään automaattisesti teeman kuvakkeena Ulkoasu-valikossa. Kuva voi olla esimerkiksi kaappaus teeman perusnäkymästä, demosivustosta tai teeman kehittäjän logosta.

3.2 Sivupohjat

Sivupohjatiedostoja (page templates) käytetään yksittäisten sivujen tai sisältötyyppien esitykseen. Teeman käyttäjien kannalta sivupohjia on kahdenlaatuisia: WordPressin standardi-sivupohjat sekä valinnaiset sivupohjat. Standardi-sivupohjia käytetään sivupohjahierarkian mukaisesti sivun sisältötyypin tai tunnisteen perusteella, jos valinnaista sivupohjaa ei ole määritelty. Taulukko 1 kuvaa tavallisimmat standardisivupohjat.

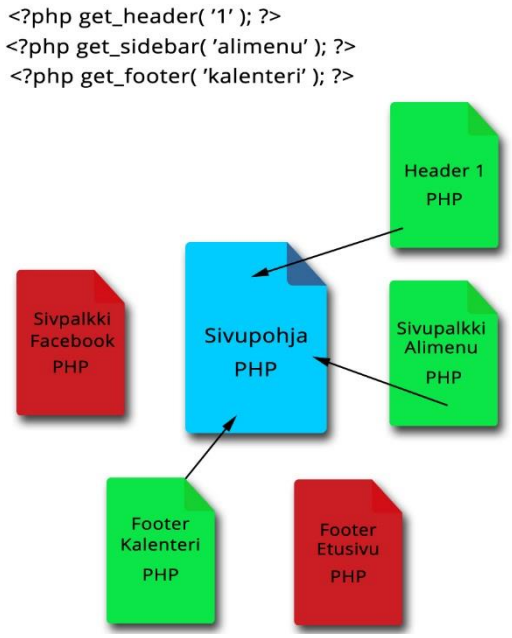
TAULUKKO 1: Luettelo yleisesti toteutetuista standardisivupohjista.

index.php	Oletussivupohja, jota WordPress käyttää viimeisenä vaihtoehtona sivun näyttämiseen, jos sisältötyypille ei löydy muita sopivia pohjia.
front-page.php	Käytetään etusivun pohjana, jos sivuston asetuksista on määritetty staattinen etusivu.
home.php	Artikkeleiden sivu, jossa on lista sivustolla olevista artikkeleista siinä tapauksessa, että sivustolle on määritetty staattinen etusivu.
page.php	Yksittäisen sivun pohja.
single.php	Yksittäisen artikkelin pohja.
singular.php	Käytetään yksittäisten sivujen, artikkeleiden ja muiden sisältötyyppien pohjana.
404.php	Käytetään virheilmoitussivuna, jos sivuston vierailija yrittää navigoida sivulle, jota ei ole olemassa.

Valinnaiset sivupohjat ovat sivuston päivittäjän käytettävissä olevia sisältöratkaisuja. Käytännössä valinnaiset sivupohjat eroavat standardi-sivupohjista siten, että niille on tiedoston kommenteissa annettu nimi Template name-rivillä. Kaikki nimetyt sivupohjat näkyvät sivuston päivittäjille valintana Sivun Ominaisuudet -valikossa sivun muokkaustilassa. Esimerkkejä valinnaiseksi toteutetuista sivupohjista ovat koko selaimen levyinen pohja tai sivupalkillinen pohja.

Sivupohjahierarkia määrittää, mitä sivupohjaa kukin sivuston sivu käyttää. Jos sivulle on asetettu valinnainen sivupohja, sitä käytetään ensisijaisesti. Valinnaisen sivupohjan puuttuessa seuraavaksi tarkistetaan löytyykö sivun polkutunnuksen mukaan nimettyä sivupohjaa. Esimerkiksi jos teemalla on tarkoitus toteuttaa yrityssivustoja, joissa yhteystiedot löytyvät omalta sivulta, voidaan tehdä sivupohja nimeltä page-yhteystiedot. Nimettyä sivupohjaa käytetään automaattisesti, kun luodaan sivu, jonka polkutunnuksena on ”yhteystiedot”. Jos polkutunnuksen mukaista sivupohjaa ei löydy, tarkistetaan löytyykö sivun ID-numeron mukaan nimettyä sivupohjaa. Page.php- ja single.php-sivupohjia käytetään näyttämään yksittäisiä sivuja ja artikkeleita, jotka eivät täytä mitään yllämainittuja ehtoja. Näiden jälkeen tulee vielä singular.php-pohja, jota käytetään, jos page.php- ja single.php-pohjia ei löydy, sisältötyypistä riippumatta. Viimeisenä vaihtoehtona käytetään index.php-sivupohjaa.

Sivupohjia luotaessa on hyvä käyttää modulaarista suunnittelua: jos jotakin sivun rakenneosaa käytetään muokkaamattomana monessa eri sivupohjassa, se kannattaa sisällyttää omaan tiedostoon, joka liitetään päätiedostoon PHP-funktioilla. Esimerkiksi sivujen ylä- ja alatunnisteet sekä mahdollinen sivupalkki on hyvä toteuttaa omiin tiedostoihinsa, koska ne on helppo liittää isäntätiedostona toimivaan sivupohjaan käyttäen tarkoitukseen luotuja WordPressin funktioita. Kuva 3 havainnollistaa kaaviolla linkitettyjen sivupohjatiedostojen suhdetta.



KUVA 3: Mallinnus linkitettyjen sivupohjatiedostojen keskinäisestä suhteesta.

Modulaarisuus myös helpottaa muokkausten tekemistä sivuston koodin. Esimerkiksi jos sivuston alatunnisteeseen on tarpeellista lisätä Googlen Analytics-palvelun seurantakoodi, muokkauksen voi tehdä footer.php-tiedostoon, jolloin kaikki sivupohjat, jotka käyttävät kyseistä tiedostoa alatunnisteen rakentamiseen, päivittyvät samalla. Kuva 4 sisältää esimerkin sivupohjan koodista. Pääsisällön lisäksi sivulla on myös sivupalkki.

```php
<?php
/**
 * The theme's index.php file.
 *
 * @package Trival
 * @subpackage Templates
 */
/* Estetään suora yhteys PHP-tiedostoihin */
if ( ! defined( 'ABSPATH' ) ) {
    exit( 'Direct script access denied.' );
}
?>
<?php get_header(); ?>                <!-- Tähän kutsutaan sivun ylätunniste erillisestä header.php -tiedostosta  -->
<div class="container">               <!-- Sivun asetteluun liittyvää HTML-koodia -->
    <div class="row">
    <div class="col-md-8">            <!-- (ALLA) Tarkastetaan onko sivulla sisältöä ja käydään se läpi silmukassa  -->
            <?php if (have_posts()) : while (have_posts()) : the_post(); ?>
                <?php the_content(); ?> <!-- Näytetään sivun sisältö -->
            <?php endwhile; else: ?> <!-- Jos sivulla ei ole sisältöä, näytetään virheilmoitus -->
            <p><?php _e('Sorry, no posts matched your criteria.'); ?></p><?php endif; ?>
    </div>
    <div class="col-md-4">

        <?php get_sidebar('sidebar_1'); ?>   <!-- Tähän kutsutaan sivupalkki erillisestä sidebar.php -tiedostosta -->

    </div>
    </div>
</div>
<?php get_footer(); ?>                <!-- Tähän kutsutaan sivun alatunniste erillisestä footer.php -tiedostosta  -->
```

KUVA 4: Koodiesimerkki sivupohjasta, jossa ylä- ja alatunnus sekä sivupalkki tulevat eri tiedostoista.

Rakenneosista pystyy myös tekemään nimettyjä versioita eri käyttötarkoituksiin. Jos on tarpeellista tehdä alatunniste, jota halutaan näyttää vain etusivulla, sen voi nimetä esimerkiksi footer-etusivu.php:ksi. Rakenneosien kutsufunktioissa voi käyttää parametrina halutun rakenneosan nimeä, tässä tapauksessa ”get_footer(’etusivu’);” etsii tiedostoa nimeltä footer-etusivu.php. Jos nimettyä tiedostoa ei löydy, käytetään nimeämätöntä footer.php:ta sen sijaan.

3.3 Lapsiteemat

Jos teemaa pitää laajentaa tai muokata yksittäistä sivustoprojektia varten, on yleinen käytäntö tehdä siitä lapsiteema. Lapsiteema perii isäntäteeman toiminnallisuuden ja tarvittaessa tyylisäännökset. Jos sivustokohtaiset muutokset toteutetaan suoraan isäntäteemaan ja teema myöhemmin päivitetään, ne saattavat kadota. Lapsiteemassa muutokset säilyvät, vaikka isäntäteemaa päivitetään.

Lapsiteemaa varten luodaan uusi alikansio WordPress-sivuston themes-kansioon. Yleisesti lapsiteema nimetään joko sivuston nimellä tai isäntäteeman mukaan lisäämällä nimeen –child-pääte. Kansioon luodaan oma style-tyylitiedosto ja useimmissa tapauksissa functions-tiedosto. Tyylitiedoston tunnisteissa ilmoitetaan isäntäteeman kansionimi Template-rivillä.

Lapsiteeman tyylitiedosto korvaa isäntäteeman tyylitiedoston: jos isäntäteeman tyylejä halutaan käyttää pohjana, täytyy sen tyylitiedosto(t) liittää lapsiteemaan erikseen. Isäntäteeman tyylitiedostot voidaan lisätä lapsiteemaan joko käyttämällä CSS:n import-lausetta lapsiteeman tyylitiedostossa, tai luomalla sen funktiotiedostoon PHP-funktio, jossa isäntäteeman tyylitiedostot ladataan käyttöön ennen lapsiteeman tyylitiedostoja.

Import-lauseen etuna on yksinkertaisuus ja useiden tyylitiedostojen ketjutusmahdollisuus: tyylitiedostossa A liitetään tyylitiedosto B, jossa liitetään tyylitiedosto C, ja niin edelleen. Funktioiden käyttö tyylitiedostojen liittämiseen on kuitenkin suositeltu toteutustapa, koska se on nopeampi sivun latauksessa (WordPress n.d.b). Tyylitiedostojen liittämistä funktioiden kautta kuvataan tarkemmin alaluvussa 5.1 Tyylitiedostot.

Tyylitiedostoista poiketen lapsiteeman funktiotiedosto ei ohita isännän funktiotiedostoa, vaan se yhdistetään niin, että isäntäteeman funktiot ladataan lapsiteeman funktioiden jälkeen. Isäntäteeman sivupohjatiedostoja voidaan kopioida lapsiteeman kansioon muokkausta varten, jolloin ne syrjäyttävät alkuperäiset tiedostot. Lapsiteeman toteutus ja muokkaus koskee enemmän itse sivustoprojekteja, mutta tekemällä valmiin pohjan, joka sisältää ainakin tyyli- ja funktiotiedostot, ja liittämällä sen osaksi pakattua teemaa, voidaan sivustoprojekteja nopeuttaa hieman lisää.

4 TEEMAN TOIMINNOT JA OHJELMOINTI

4.1 Funktiot

Suurin osa teemojen ja lisäosien toiminnallisuudesta toteutetaan PHP-funktioilla. Lähtökohtana toimii functions.php-tiedosto, johon yleisesti toteutetaan ainakin rakenteellisesti oleelliset toiminnot kuten valikkojen lataus. Kaikkien sivuston käytössä olevien lisäosien ja teeman funktioiden nimien pitää olla yksilöllisiä. Koska teeman kehittäjä ei pysty vaikuttamaan sivustolle ladattaviin lisäosiin, kannattaa funktioiden ja mahdollisten luokkien nimissä käyttää esimerkiksi teeman nimeä tai tekstitunnusta etuliitteenä ristiriitojen välttämiseksi.

Teemasta saatetaan tehdä lapsiteemoja, joissa pitää muokata isäntäteeman funktioita. Funktion kopioiminen suoraan lapsiteeman funktiotiedostoon muokkausta varten johtaa sivuston virhetilaan, koska kahta samanimistä funktiota ei saa olla sivuston koodissa. Tästä syystä isäntäteeman funktiot kannattaa toteuttaa jo kehitysvaiheessa ohitettaviksi lisäämällä funktion ympärille if-lauseke, joka tarkistaa onko kyseinen funktio jo esitelty lapsiteeman funktiotiedostossa.. Kuvassa 5 on esimerkki, miten tarkistus voidaan toteuttaa.

```
/* Jos funktiota trival_custom_logo_setup ei ole aiemmin määritelty, määritellään se nyt */
if ( ! function_exists( 'trival_custom_logo_setup' ) ) {
    function trival_custom_logo_setup() {
        $defaults = array(
        'height'      => 100,
        'width'       => 400,
        'flex-height' => true,
        'flex-width'  => true,
        'header-text' => array( 'site-title', 'site-description' ),
        );
        add_theme_support( 'custom-logo', $defaults );
    }
}
add_action( 'after_setup_theme', 'trival_custom_logo_setup' );
```

KUVA 5: Esimerkki funktioista, jonka voi korvata esimerkiksi lapsiteeman funktiotiedostossa rikkomatta sivustoa.

Myös JavaScriptiä voidaan käyttää teemassa. JavaScriptin ohjelmointiosuuden voi toteuttaa suoraan sivupohjatiedostoihin, PHP-funktiotiedostoihin tai omiin tiedostoihinsa käyttötarkoituksen mukaan. Jos teemaan toteutetaan joukko JavaScript-funktioita, joita kutsutaan toistuvasti eri sivuilla tarpeen mukaan, voidaan ne lisätä samaan tiedostoon, joka ladataan selaimen muistiin ylätunnisteen sivupohjatiedostossa.

4.2 Koukut

Koukut (hooks) ovat tapa muokata WordPressin suoritusajoa ja liittää teemojen ja lisäosien funktioita WordPressin omiin funktioihin. Koukkujen käyttö teemojen ja lisäosien kehityksessä on välttämätöntä, jotta ne saadaan liitettyä WordPressiin ilman, että WordPressin ydintiedostoja muokataan.

Jokaisen omatekoisen funktion yhteydessä toteutetaan myös funktion liitos koukkuun, jolla funktion suoritus tapahtuu, poikkeuksena funktiot, joita kutsutaan toisista teeman tai lisäosan funktioista tai joita käytetään omissa luokissaan. Koukutettavat funktiot jaetaan kahteen pääryhmään niiden käyttämien koukkujen ja toimintaperiaatteen mukaan: toiminnot (actions) ja suodattimet (filters).

Toiminnoilla tarkoitetaan teeman tai lisäosan funktioita, jotka liitetään toimintakoukkuihin toteutumaan tietyssä vaiheessa sivuston latausta, useimmiten osana WordPressin ydinprosesseja. WordPressin omia toimintakoukkuja on satoja. Teemakehityksen kannalta hyödyllisiä koukkuja ovat muun muassa wp_enqueue_scripts, jolla liitetään CSS- ja JavaScript-tiedostoja sivustolle. Toinen hyödyllinen koukku on after_theme_setup, jota usein käytetään ilmaisemaan teeman tuki tietyille WordPressin tai sen lisäosien ominaisuuksille ja lisäosille.

Suodattimilla tarkoitetaan funktioita, jotka liitetään suodatinkoukkuihin niin, että ne muokkaavat muiden funktioiden käsittelemää tietoa puuttumatta muuten niiden toimintaan. Kirosanasuodatin on nimensä mukaisesti klassinen esimerkki, jonka voi toteuttaa suodattimella, joka liitetään sivuston käyttäjien kommentteihin poistamaan niissä esiintyvät kirosanat ennen kommenttien tallentamista tietokantaan. Myös myöhemmin opinnäytetyössä kuvattavien lyhytkoodien toteutus tapahtuu WordPressin omalla tarkoitukseen tehdyllä suodattimella.

Toiminnoilla ja suodattimilla voidaan saada aikaan samanlaisia lopputuloksia eri toteutustavoilla. Teeman funktioihin voi myös toteuttaa omia koukkuja, joita lisäosat tai teeman laajennukset voivat käyttää. Myös koukkujen nimeämisessä kannattaa käyttää etuliitettä ristiriitojen välttämiseksi.

4.3 Teeman lokalisaatiotuki

Jollei teemaa ole tarkoitettu käytettäväksi vain yhden yksikielisen sivuston toteutukseen, tulee sen kielikääntäminen todennäköisesti jossain vaiheessa ajankohtaiseksi. Tämän takia teema kannattaa alusta asti kehittää tukemaan kielikäännöksiä, jottei teeman sisäisiä tekstejä tarvitse myöhemmin käydä muuttamassa.

WordPress käyttää vapaan lähdekoodin gettext-kirjastoja ja -työkaluja kääntämiseen. Teemojen ja lisäosien kehittäjät merkitsevät lähdekoodissa käytetyt merkkijonot käännöstä varten, ne jäsennetään gettext-työkaluilla ja tallenetaan POT-tiedostoiksi (portable objects template). POT-tiedostoissa olevat merkkijonot käännetään joko lokaalisti tai WordPressin verkkoyhteistyökalu GlotPressin kautta halutulle kielelle ja lopulta binäärimuotoisiksi MO-tiedostoiksi. (WordPress n.d.c).

Merkkijonojen merkitseminen käännettäväksi tapahtuu sisällyttämällä ne WordPressin käytössä oleviin gettext-funktioihin. Merkkijonon ja muiden mahdollisten lisäparametrien lisäksi funktion parametrina ilmoitetaan tekstitunnus, joka merkitsee, että merkkijono on osa teeman yhtenäistä käännöstiedostoa. Tekstitunnuksen on oltava sama kuin teeman style.css-tiedostossa ilmoitettu tekstitunnus. Käytettävillä eri gettext-funktioilla voi muun muassa tukea tekstissä olevia muuttujia tai välittää kääntäjille kommentteja tekstistä, kuten sanan tai ilmaisun asiayhteyden. Kuvassa 6 on esimerkki staattisten merkkijonojen merkitsemisestä käännettäviksi WordPressin __()-funktiolla. Itse käännösprosessia ei käydä läpi tässä opinnäytetyössä.

```
'label'              => __( 'Header Logo Align', 'trival-theme' ),
'description'        => __( 'Align header logo position','trival-theme' ),
```

KUVA 6: Merkkijonojen toteutus lokalisaatiotuella.

4.4 Bootstrap

Bootstrap on ilmainen ja vapaan lähdekoodin web-ohjelmistokehys, jota käytetään responsiivisten ulkoasurakenteiden toteutukseen verkkosivuilla ja –applikaatioissa. Alun pe-

rin Twitterin ohjelmistokehyksenä kehitetty Bootstrap keskittyy ainoastaan julkisen puolen esittämiseen. Bootstrapin uusin versio 4.0 on saatavilla osoitteessa: http://getbootstrap.com/

Bootstrap-tiedostot lisätään teeman alle omiin kansioihinsa ja liitetään sivuston koodiin WordPressin koukuilla. Sivuston HTML-rakenne toteutetaan Bootstrapin käyttämän luokkajärjestelmän mukaisesti, jotta sivuston responsiivisuus toimii. Bootstrap-integraation toteutukseen löytyy opetusmateriaalia runsaasti verkosta, tässä opinnäytetyössä ei keskitytä siihen.

Bootstrapin käytöstä WordPress-teemoissa löytyy eriäviä mielipiteitä. Bootstrap helpottaa responsiivisten sivustojen kehitystä ja tarjoaa valmiina monia modernien sivustojen graafisia ominaisuuksia, kuten animaatioita, vihjelaatikoita, nappeja ja ruudun leveyden mukaan vaihtuvan valikkorakenteen. Kaikkien ominaisuuksien liittäminen teemaan vaatii kuitenkin opettelua ja hidastaa sivuston latausta, ellei tarpeettomia ominaisuuksia poisteta koodista. Bootstrapin käyttö saa myös kritiikkiä luovuuden tukahduttamisesta verkkosivujen suunnittelussa juuri edellä mainittujen valmiiden ominaisuuksien yleistyneen käytön takia, ilman yksilöllisiä muokkauksia. (McCollin 2015.)

Tässä opinnäytetyössä käytetään Bootstrapia, koska se tarjoaa yksinkertaisen lähtökohdan responsiivisten verkkosivujen toteutukseen. Myöhemmin saattaa tulla ajankohtaiseksi korvata Bootstrap muulla, mahdollisesti omalla ulkoasukehyksellä.

5 TEEMAN ULKOASU JA MUKAUTUS

5.1 Tyylitiedostot

Tyylitiedostoilla rakennetaan teeman perusulkoasu, jonka päälle sivustoprojektin yksilöllinen ilme toteutetaan. Teeman päätyylitiedostona toimii style.css, joka yllä mainitun mukaisesti on pakollinen osa teeman rakennetta, vaikka se ei sisältäisi muuta kuin ylätunnisteen kommenttiosion. Oletuksena se sisältää kaikki teeman tyylisäännökset, mutta teeman toteutustavan mukaan säännöksiä voidaan jakaa useampaan tyylitiedostoon alueittain. Tyylitiedostot liitetään teemaan käyttämällä wp_enqueue_style-funktiota, jonka parametreina annetaan tyylitiedostolle vapaavalintainen, mutta uniikki tunnistenimi - sekä hakemistopolku, josta tiedosto löytyy.

Vapaavalintaisista lisäparametreista huomionarvoinen on myös riippuvuussuhteen määrittäminen. Riippuvuussuhteella pystytään tarpeen mukaan vaikuttamaan, mitkä tyylisäännöt korvaavat toisensa päällekkäisyystapauksissa. Esimerkiksi jos style.css-tiedostoon halutaan toteuttaa lisäyksiä, jotka korvaavat muiden tiedostojen sisältämiä tyylisääntöjä, voidaan style.css asettaa latautumaan viimeiseksi määrittelemällä riippuvuussuhde muihin tyylitiedostoihin, kuten kuvassa 7. Täten style.css:n sisältämät tyylisäännöt ohittavat muissa tiedostoissa olevat ristiriitaiset tyylisäännöt.

```
function theme_styles() {
    wp_enqueue_style( 'bootstrap_css', get_template_directory_uri() . '/bootstrap/css/bootstrap.css'  );
            //wp_enqueue_style()-funktiolle ilmoitetaan tyylitiedostosta käytettävä uniikki nimi ja tiedoston polku.
    wp_enqueue_style( 'main_css', get_template_directory_uri() . '/style.css', array( 'bootstrap_css' ) );
            //Funktiolle voi myös asettaa valinnaisena parametrina taulukon, jossa on ensin ladattavien tyylitiedostojen nimet.
            //Tällöin kyseistä tyylitiedostoa ei ladata ennenkuin taulukossa olevat tyylitiedostot on käsitelty.
}
add_action( 'wp_enqueue_scripts', 'theme_styles');  //Koukku, jolla tyylitiedostojen latausfunktio liitetään WordPressin ajoon.
```

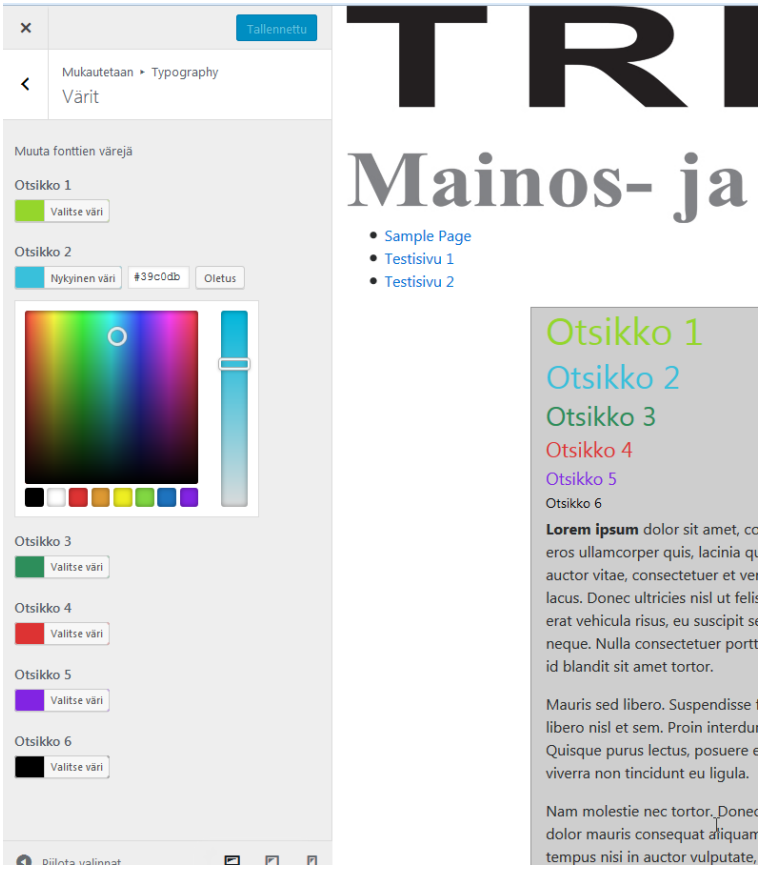
KUVA 7: Funktio, jolla tyylitiedostot liitetään WordPressiin.

5.2 Teeman mukautus

Teeman rakennetta ja ulkoasua voi muokata sivustoprojektiin sopivaksi tyylitiedostoa muokkaamalla, mutta huomattavasti käyttäjäystävällisempi ja teeman uudelleenkäytön kannalta parempi ratkaisu on toteuttaa teemalle asetusvalikko. Asetusvalikossa sivuston kehittäjät tai ylläpitäjät voivat esimerkiksi määrittää sivustolla käytettävät fontit ja kirjasinkoot sekä sivuston värimaailman ilman ohjelmointia. Sivuston koodin mukauttaminen

asetuksissa määriteltyihin arvoihin tuo omat haasteensa teeman kehitykseen, mutta pitkällä tähtäimellä se säästää aikaa ja vaivaa, koska sivustoprojektien toteutus nopeutuu huomattavasti.

Teeman asetusvalikon toteutukseen on käytössä kaksi tapaa. Alun perin ainoana vaihtoehtona oli sivuston ohjausnäkymään itse toteutettava valikkosivu, joka on yleisesti nimetty Teeman asetuksiksi (Theme Options) ja joka sijaitsee Ulkoasu-valikon alla. WordPressin versiossa 3.4 julkaistu Mukauta-näkymä (Customizer API) tarjoaa toisen vaihtoehdon, jota WordPressin kehitystiimi suosittelee nykyisin käytettäväksi nimenomaan teeman asetusten muokkaukseen. Tässä opinnäytetyössä keskityttiin asetusvalikon toteutukseen Mukauta-näkymässä (kuva 8).



KUVA 8: Mukauta-näkymä ja esimerkkinä toteutettu otsikoiden värivalikko. Muutokset päivittyvät suoraan oikealla olevaan esikatselunäkymään.

Mukauta-näkymän suurimpana etuna on esikatselumahdollisuus: teeman asetuksiin tehdyt muutokset päivittyvät välittömästi sivun esikatselunäkymään, mutta eivät näy julkisesti ennen tallennusta. Mukauta-näkymässä on myös mahdollista navigoida sivulta toiselle ennen muutosten tallennusta.

Mukautettavat ominaisuudet koostuvat asetuksista ja kontrollereista. Asetus on mukautettavan ominaisuuden arvo, jota käytetään sivuston muokkaukseen. Kontrolleri on Mukauta-näkymän osa, jolla teeman käyttäjä voi muuttaa asetusta. Kontrolleri voi olla esimerkiksi tekstikenttä, liukukytkin, värivalikko tai valintaruutu.

Uusi sisältö ja toiminnallisuus lisätään Mukauta-näkymään itsetehdyllä funktiolla, joka liitetään customize_register-koukulla WordPressin ajoon. Funktiolle välitetään parametrina WP_Customize_Manager()-luokan objekti, jonka omia metodeja kutsutaan paneelien, osastojen, asetuksien ja kontrollerien lisäykseen, muokkaukseen ja poistamiseen. Asetuksille ja Mukauttimen osille on annettava yksilölliset nimet, joilla ne viittaavat toisiinsa Mukauta-näkymän rakennetta muodostettaessa. Lisäksi asetuksien arvoa haetaan nimen perusteella myöhemmässä käytössä.

Asetuksille voi myös asettaa oletusarvon, joka näytetään kontrollerissa ennen kuin käyttäjä on muokannut asetusta. Lisäksi asetuksille suositellaan toteutettavaksi myös siistimisfunktio (sanitize_callback), joka käsittelee käyttäjän syöttämät arvot mahdollisten virheiden varalta. Injektiohyökkäyksiin ja muihin mahdollisiin väärinkäyttöihin ei liene tarpeellista puuttua siistimisen yhteydessä. Mukauta-näkymään pääsy vaatii pääkäyttäjän oikeudet, joilla voi muutenkin käsitellä sivuston tietokantoja. Kuvassa 9 on esimerkki asetuksen määrittämisestä Mukauttimeen.

```
add_action('customize_register','trival_theme_customize_register');
function trival_theme_customize_register( $wp_customize ) {

        global $default_values;

        if ( class_exists( 'WP_Customize_Control' ) ) {
            require_once 'class-customizer-range-value-control.php';
        }

        /* LOGO TOP MARGIN */
        $wp_customize->add_setting( 'logo_margin_top_setting', array(
        'default'           => $default_values["logo_margin_top_default"],
        'type'              => 'theme_mod',
        'transport'         => 'refresh',
        'sanitize_callback' => 'trival_sanitize_pixel_range',
        ) );
```

KUVA 9: Esimerkki uuden asetuksen lisäyksestä Mukauta-näkymään. Mukana oletusarvon ja siistimisfunktion asetus.

Mukauttimen asetusten arvoja voidaan kutsua toisaalla sivuston koodissa get_theme_mod()-funktiolla, jonka parametrina käytetään asetuksen nimeä ja valinnaisesti oletusarvoa, jos käyttäjä ei ole vielä muokannut asetusta. Huomioitavaa on, että aiemmin mainittu asetuksen lisäyksessä Mukauttimeen ilmoitettu oletusarvo ei tallennu varsinaisen asetuksen arvoksi missään vaiheessa. Se näkyy vain asetuksen kontrollerissa Mukauta-näkymässä. Jos asetusta ei ole muokattu, eikä get_theme_mod()-kutsussa käytetä oletusarvoa lisäparametrina, palauttaa kutsu tyhjää, mikä saattaa johtaa virhetilanteisiin koodissa.

5.2.1 Sisällön luonti Mukauta-näkymällä

Mukauta-näkymän asetuksien perusteella voidaan muokata myös sivuston sisältöä. Esimerkiksi sivuston yläosaan voidaan header.php-tiedostossa toteuttaa kohta, jossa haetaan ja näytetään ylläpitäjän Mukauta-näkymän asetuksiin syöttämät yhteystiedot. Yksinkertaisimmillaan se tapahtuu kuvan 10 mallin mukaisesti.

KUVA 10: Yhteystietojen tuominen mukauttimen asetuksista. Alla koodi, jolla asetuksista haetaan mukautettu arvo.

Esimerkissä ylläpitäjä joutuu itse syöttämään mahdolliset mailto- tai tel-URI-tunnisteet, joilla yhteydenotto annettuihin sähköpostiosoitteisiin tai puhelinnumeroihin helpottuu. Tämän voi tehdä myös lisäämällä HTML-koodit ja URI-tunnisteet valmiiksi sivupohjaan funktiokutsun ympärille, jolloin ylläpitäjän tarvitsee vain syöttää pelkkä puhelinnumero tai sähköpostiosoite kyseiseen kenttään. Haittapuolena on, että tämän jälkeen sivuston ylläpitäjä ei voi käyttää kyseistä kenttää muun sisällön näyttämiseen.

Myös sivupohjien rakenteen pystyy mukauttamaan esimerkiksi luomalla if-lausekkeen, jossa tarkistetaan, onko jokin tietty asetus voimassa. Kuvassa 11 on toteutettu sivupohja, joka sisältää yksinkertaisen tarkistuskoodin siitä, kummalle puolelle pääsisältöä sivupalkki tulee. Kuvan koodissa tarkistus on esimerkin vuoksi toteutettu huonosti. Jos käyttäjä ei ole Mukauta-näkymässä tallentanut jompaakumpaa vaihtoehdoista, palauttaa get_theme_mod()-funktio tyhjää, koska oletusarvoa ei ilmoiteta kummassakaan kutsussa. Tällöin kumpikaan ehdoista ei toteudu ja sivupalkki jää kokonaan puuttumaan. Tässä tapauksessa yksinkertaisin korjaus on yhdistää if-lauseet niin, että jälkimmäinen osa korvataan else-osalla, jolloin sivupalkki tulee aina oikealle, paitsi jos asetuksen arvo on ”left”.

```php
<?php get_header(); ?>
<div class="container">
    <div class="row">

        <!-- Jos sivupalkin sijainniksi on asetuksissa määritelty 'left', sijoita se ennen sisältöä -->
        <?php
        if ( get_theme_mod( 'theme_sidebar_location' ) == 'left' ) {
            echo '<div class="col-md-4">';
            <?php get_sidebar('sidebar_1');
            echo '</div>';
        } ?>

        <!-- Sisältö -->
        <div class="col-md-8">
            <?php if (have_posts()) : while (have_posts()) : the_post(); ?>
                <?php the_content(); ?>
            <?php endwhile; else: ?>
            <p><?php _e('Sorry, no posts matched your criteria.'); ?></p><?php endif; ?>
        </div>

        <!-- Jos sivupalkin sijainniksi on asetuksissa määritelty 'right', sijoita se sisällön jälkeen -->
        <?php
        if ( get_theme_mod( 'theme_sidebar_location' ) == 'right' ) {
            echo '<div class="col-md-4">';
            <?php get_sidebar('sidebar_1');
            echo '</div>';
        } ?>
    </div>
</div>
<?php get_footer(); ?>
```

KUVA 11: Tarkistetaan if-lauseella onko sivupalkin paikka määritelty Mukauttimen asetuksissa vasemmalle vai oikealle puolelle sivua.

5.2.2 Sivuston tyylitys Mukauta-näkymällä

Teeman kehittäjä pystyy lisäämään Mukauta-näkymään käytännössä kaikki sivuston oleelliset CSS-ominaisuudet, joihin teeman käyttäjä voi syöttää haluamansa arvot. Nämä arvot voidaan muuttaa dynaamisesti tyylisäännöiksi ja lisätä sivuston ulkoasuun muutamalla eri tavalla.

Voidaan esimerkiksi toteuttaa PHP-tiedosto, joka on samalla tyylitiedosto. Tällöin voidaan käyttää get_theme_mod()-kutsuja suoraan hakemaan Mukauttimessa säädettyjä arvoja tyylisäännön ilmoituksessa. Vaihtoehtoisesti arvot voidaan tallentaa muuttujiin, joita sitten käytetään ilmoituksessa. Ulkoisen, dynaamisen tyylitiedoston etuina ovat selkeys teeman toteutusvaiheessa sekä sivujen HTML-koodin siisteys, joka parantaa sivuston hakukonenäkyvyyttä jonkin verran. Kuvassa 12 on havainnollistava esimerkki dynaamisesta tyylitiedostosta.

```php
<?php
//Ilmoitetaan tiedoston sisältötyyppi
header("Content-type: text/css; charset: UTF-8");

//Ilmoitetaan oletusarvo, jota käytetään jos käyttäjä ei ole tallentanut Mukauta-näkymässä mitään.
$backgroundColorDefault = "#ffffff";

//Haetaan Mukauttimessa oleva arvo ja ilmoitetaan varuiksi oletusarvo, jos Mukauttimen asetuksen arvo on tyhjä
$backgroundColor = get_theme_mod('background_color', $backgroundColorDefault);
?>

body {
    /* Käytetään yllä luotua muuttujaa tyylisäännöksen ilmoituksen arvona */
    background-color: <?php echo $backgroundColor; ?>;
}
```

KUVA 12: Dynaaminen tyylitiedosto, jossa käytetään sekä PHP:ta että CSS-kieltä.

Toinen tapa on käyttää WordPressin wp_add_inline_style()-funktiota, jolla voidaan välittää merkkijonona CSS-tyylisäännöksiä lisättäväksi sivun sisäiseen tyylitykseen. Funktion nimestä huolimatta funktio ei lisää tyylisäännöksiä suoraan yksittäisiin HTML-elementteihin, vaan kootusti sivun head-osioon. Merkkijonon tyylisäännöistä voi rakentaa esimerkiksi funktiolla, joka saa parametrina kaksiulotteisen taulukon. Uloin taulukko koostuu avain-arvo-pareista, joissa avain on CSS-selektori, jolle tyylisäännöt osoitetaan ja arvo on sisempi taulukko. Sisempi taulukko koostuu avain-arvo-pareista, joissa avain on muokattava ominaisuus ja arvo on mukautetun asetuksen arvo. Kuvassa 13 on esimerkin mukaan toteutettu funktio. Lyhyt esimerkki parametrina välitetystä kaksiulotteisesta taulukosta: "body" -> ("background-color" -> ASETUKSESTA_HAETTU_VÄRI, "font-size" -> ASETUKSESTA_HAETTU_FONTIN_KOKO).

```php
//Funktio, jossa CSS-tyylisäännökset rakennetaan merkkijonona kaksiulotteisesta taulukosta.
function custom_update_css_array($style_array) {

    $css = ''; //Aloitetaan tyhjästä merkkijonosta.

    foreach ( $style_array as $key => $inner_style_array ) { //Käydään läpi uloin taulukko.

        $css .= $key . '{'; //Merkkijonoon lisätään uloimman taulukon avain, joka on nimetty muokattavan
                            //elementin mukaan, sekä avataan ilmoituslohko.

        foreach ( $inner_style_array as $attribute => $value ) { //Käydään läpi sisäiset taulukot.
            $css .= $attribute . ':' . $value . ';'; //Lisätään sisäisen taulukon avain-arvo-parit merkkijonoon.

        }

        $css .= '}'; //Suljetaan ilmoituslohko
    }

    //Lisätään valmis merkkijono sivustoon.
    wp_add_inline_style( 'customizer_css', $css );
}
```

KUVA 13: Sisäisen CSS-tyylityksen lisäys teemaan.

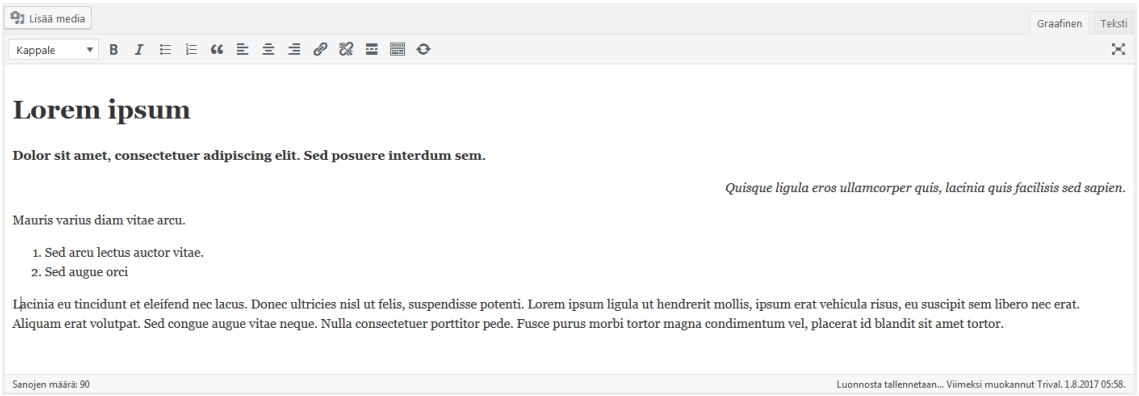
Sisäisen tyylityksen etuna on, että se on helpompi kohdistaa tietyille sivuille tai sivupohjille. Lisäksi tyylisäännöksinä toimivat merkkijonot on helppo rakentaa funktioissa ilman

välimerkkejä ja rivinvaihtoja. Näin tyylisäännökset ovat automaattisesti minimoituja. Minimoiminen tarkoittaa välilyöntien ja rivinvaihtojen poistoa koodista, mikä pienentää kooditiedostojen kokoa, mutta samalla tekee koodista vaikealukuisempaa ihmissilmälle.

6 WORDPRESS-MUOKKAUSNÄKYMÄN LAAJENTAMINEN

6.1 Visuaalinen muokkain

WordPress-sivustoilla käytetään sisällöntuottamiseen TinyMCE-nimistä visuaalista muokkainta, joka toimii WYSIWYG-periaatteella. WYSIWYG on lyhennys sanoista What You See Is What You Get, suomennettuna mitä näet, sitä saat. Tämä tarkoittaa, että tuotettu sisältö näyttää muokkaimessa lähes samalta (kuitenkin ilman teeman tyylisäännöksiä) kuin julkaistuna sivustopuolella. Sisällönmuodostuksessa käytetty HTML-koodi ei näy graafisessa näkymässä, vaan erillisessä tekstinäkymässä. Kuvassa 14 on visuaalisen muokkaimen perusnäkymä. Lisää media -napista voi tekstiin lisätä kuvia tai muita tiedostoja. Oikean yläkulman tekstivälilehdestä voi muokata sisällön HTML-koodia.



KUVA 14: Kuva TinyMCE-muokkaimen perusnäkymästä.

Visuaalinen muokkain sisältää oletuksena tavallisimpia HTML-koodilla toteutettavia tekstin ja kuvien muokkaukseen soveltuvia toimintoja. Näitä ovat esimerkiksi tekstin lihavointi, värjäys, otsikointi ja tasaus. Tekstiin voi myös lisätä linkkejä helppokäyttöisen ponnahdusvalikon kautta. Muokkain myös lisää automaattisesti kappalejaot ja rivinvaihdot sivun koodiin. Täysin ongelmaton muokkain ei ole: tekstiä muokatessa HTML-elementtien sulkemiskoodit voivat eksyä väärään paikkaan ja aiheuttaa poikkeavuuksia sivun julkisessa näkymässä.

6.2 Lyhytkoodit

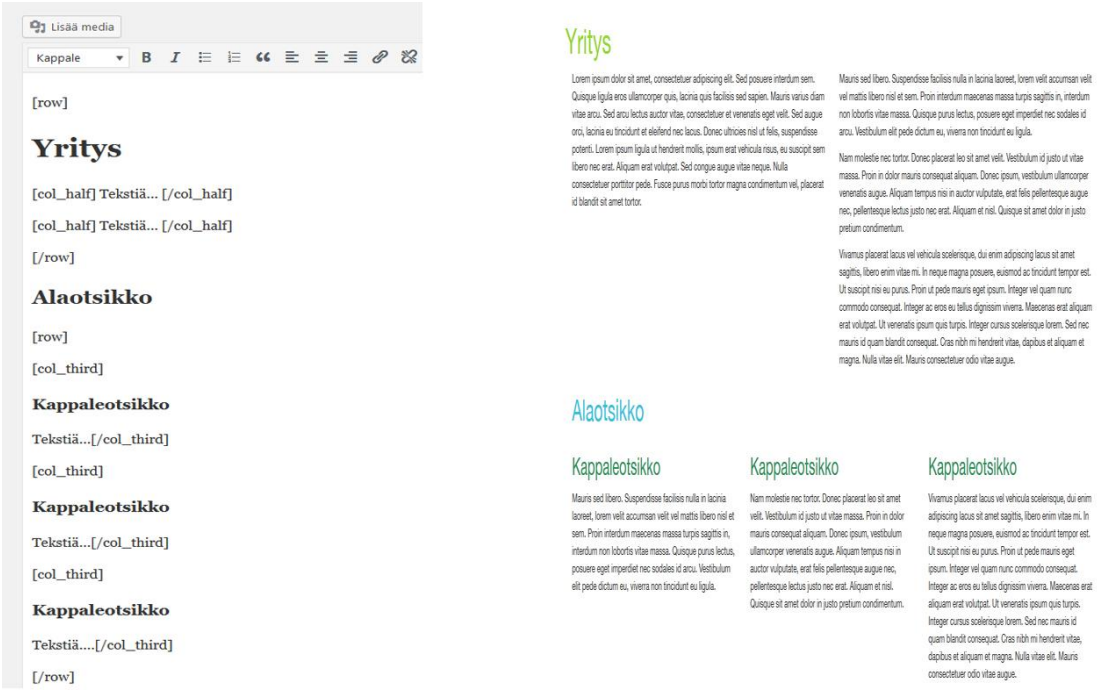
Lyhytkoodit ovat WordPressin ominaisuus, jolla sivuston päivittäjä pystyy lisäämään sivulle, artikkeliin tai tekstivimpaimeen monimutkaisia sisältöelementtejä ja koodinpätkiä lyhyillä tekstimakroilla. Lyhytkoodin käyttö ei edellytä HTML-koodin tuntemusta ja WordPressin visuaalinen muokkausnäkymä säilyy helppolukuisempana.

Lyhytkoodit toteutetaan sivuston koodiin PHP-funktiolla, joka sisältää lyhytkoodin toiminnallisuuden. Funktio rekisteröidään WordPressin käyttöön add_shortcode-koukulla, jonka jälkeen tekstissä esiintyvät lyhytkoodit korvataan automaattisesti funktion sisällöllä. Lyhytkoodien kautta on myös mahdollista välittää funktiolle parametreja, joilla funktion käyttäytymistä ja lopputulosta voi mukauttaa. Lopuksi funktio yleisesti palauttaa HTML-muodossa lyhytkoodin toiminnallisuuden, joka näytetään lyhytkoodin tilalla sivun tai artikkelin julkisessa näkymässä.

Lyhytkoodit voivat olla itsestäänsulkeutuvia, jolloin ne koostuvat vain yhdestä tunnisteesta hakasulkeiden sisällä, tai suljettavia, jolloin ne tarvitsevat avaus- ja lopetustunnisteet. Lyhytkoodifunktiot ohjelmoidaan vastaanottamaan kaksi parametria. Ensimmäisenä ovat lyhytkoodin valinnaiset attribuutit, jotka välitetään lyhytkoodissa funktiolle HTML-elementtien attribuuttien mukaisesti: [lyhytkoodi attribuutti=”arvo”]. Toisena parametrina on lyhytkoodi-tunnisteiden välinen sisältö, jos lyhytkoodi on suljettava. Sisältö liitetään funktion lopussa olevaan palautukseen.

Opinnäytetyössä kehitettävään teemaan toteutetaan lyhytkoodeja, joilla teeman käyttäjä pystyy muokkaamaan sisällön asettelua käyttämällä Bootstrapin ruudukkojärjestelmää jakamaan sivuston sisältöä riveihin ja sarakkeisiin. Ruudukkojärjestelmää on mahdollista käyttää lisäämällä muokkaimen tekstinäkymään HTML-div-elementtejä, jotka käyttävät Bootstrapin CSS-luokkia, mutta automatisoimalla HTML-koodin lisäys tekstiin lyhytkoodeilla parannetaan teeman helppokäyttöisyyttä.

Rivien ja sarakkeiden avausta ja sulkemista varten toteutetaan row- ja col-lyhytkoodit. Bootstrapissa yhden rivin voi jakaa maksimissaan kahteentoista sarakkeeseen: teemaan käytettävyyden kannalta on hyvä toteuttaa lyhytkoodit ainakin puolikkaita ja kolmas- ja neljäsosajakoja varten. Kuvassa 15 havainnollistetaan lyhytkoodin toiminta WordPressin sivumuokkaustilassa. Tekstikappaleet ovat muokkaimessa lyhennetty luettavuuden helpottamiseksi. Oikealla näkyy, miltä teksti näyttää sivuston julkisella puolella.



KUVA 15: Lyhytkoodin toiminta visuaalisessa muokkaimessa.

Yksinkertaisimmillaan rivi- ja sarakekoodien toteutus tapahtuu luomalla funktiot, joille välitetään sisältönä avaus- ja sulkemistunnisteiden välissä oleva teksti. Tämän sisällön ympärille lisätään div-elementti, jonka luokkana on tarkoitusta vastaava Bootstrap-luokka, esimerkiksi row-luokka rivinmuodostusta varten. Lopuksi funktio palauttaa muokatun sisällön. Funktiot ympäröivät parametrina saadun tekstisisällön div-elementeillä, joilla sarakejaottelu tapahtuu ja palauttavat sen. Kuvassa 16 on malli lyhytkoodin PHP-funktion toteutuksesta.

```
/*
 *      ROW
 */

// Generate DIV with row class
function trival_add_row_shortcode( $atts, $content = null ) {

    // Return Row Content
    return '<div class="row">' . shortcode_unautop( do_shortcode($content) ) . '</div>';

}
add_shortcode( 'row', 'trival_add_row_shortcode' );

/*
*       HALF
*/

// Generate 1/2 column
function trival_add_column_1_half_shortcode( $atts, $content = null ) {

    // Return Column Content
    return '<div class="col-md-6">' . shortcode_unautop( do_shortcode($content) ) . '</div>';

}
add_shortcode( 'col_half', 'trival_add_column_1_half_shortcode' );
```

KUVA 16: Lyhytkoodin toiminnallisuuden toteutus.

Oletuksena WordPress ei tue sisäkkäisiä lyhytkoodeja, vaan lyhytkoodin sisällön lyhytkoodit palautuvat tavallisena tekstinä. Jotta myös sisäkkäiset lyhytkoodit suoritetaan, täytyy sisältö käsitellä WordPressin do_shortcode()-funktiolla välittämällä sisältö parametrina. Lisäksi sisäkkäiset lyhytkoodit saattavat aiheuttaa ongelmia sivuston HTML-koodissa, koska WordPress lisää automaattisesti kappale-elementit tekstiin ja ne voivat mennä ristiin lyhytkoodeissa tehtyjen elementtien kanssa. Tätä varten kannattaa tehdä funktio, joka käsittelee kaikki sisäkkäiset lyhytkoodit ja poistaa ylimääräiset kappale-elementit lyhytkoodien ympäriltä.

6.3 Visuaalisen muokkaimen laajennus

TinyMCE-muokkaimen työkalurivistä löytyy oletuksena tavallisimpia tekstinmuokkausvälineitä otsikoinnista sisennyksiin. Muokkaimeen on kuitenkin mahdollista lisätä myös omia painikkeita ja alasvetolaatikoita. Esimerkiksi yllä toteutetut ruudukkolyhytkoodit voidaan lisätä työkaluriviin käyttäjäystävällisyyden parantamiseksi. Opinnäytteen teemaan toteutettiin painike, jolla käyttäjä voi valita sarakejaon, joka lisätään tekstiin. Alasvetolaatikon valinta lisää automaattisesti rivi- ja sarakelyhytkoodit käyttäjän valinnan mukaan. Laajennukset muokkaimeen toteutetaan TinyMCE:n lisäosina, jotka voidaan lisätä osaksi teeman rakennetta.

Lisäosa ja sen lisäämät painikkeet rekisteröidään WordPressiin käyttämällä tarkoitukseen luotuja suodattimia kuvan 17 mukaisesti. Suodattimet liitetään toiminta-koukulla WordPressin suoritusvaiheeseen. Ensimmäinen suodatin, mce_external_plugins, liittää uuden lisäosan ja sen JavaScript-koodin osaksi TinyMCE-muokkainta. Toinen suodatin, mce_buttons, lisää painikkeet osaksi muokkaimen työkaluriviä.

```
/* Toimintafunktio ajoittaa suodatinfunktioiden toteutuksen WordPressin käynnistykseen */
add_action( 'init', 'trivalmce_buttons' );
function trivalmce_buttons() {
    add_filter( "mce_external_plugins", "trivalmce_add_buttons" );
    /* mce_buttons-suodatin liittää painikkeet osaksi työkaluriviä. Suodattimena pystyy myös käyttämään esimerkiksi
    * mce_buttons_2, jolloin painikkeet sijoittuvat toiselle riville.
    */
    add_filter( 'mce_buttons', 'trivalmce_register_buttons' );
}

/* Suodatinfunktio, joka liittää tehtävän lisäosan ja sen JavaScript-koodin osaksi TinyMCE-muokkainta */
function trivalmce_add_buttons( $plugin_array ) {
    $plugin_array['trivalplugin'] = get_template_directory_uri() . '/js/trival_tinymce_buttons.js';
    return $plugin_array;
}

/* Suodatinfunktio, joka liittää painikeet osaksi  TinyMCE:n työkaluriviä */
function trivalmce_register_buttons( $buttons ) {
    array_push( $buttons, 'addRowButton' );
    return $buttons;
}
```

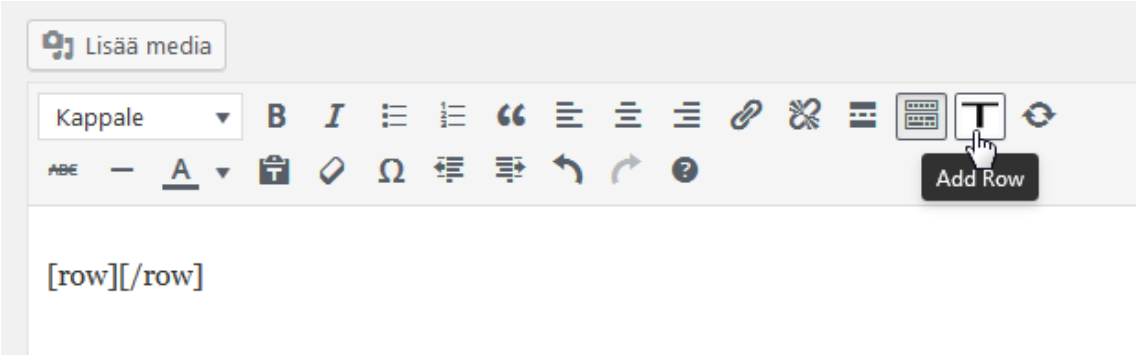
KUVA 17: TinyMCE:n laajennuksien rekisteröinti WordPressiin.

Painikkeiden luonti ja toiminnallisuuden toteutus tapahtuu JavaScript-koodilla, joka on hyvä tehdä omaan tiedostoon. Koodissa määritellään funktio, jonka suorituksen yhteydessä luodaan uudet painikkeet addButton-koodilla. Painikkeille määritellään nimi, valinnainen kuvake ja toiminto, joka suoritetaan, kun painiketta painetaan. Painikkeen toiminto lisätään addCommand-koodilla, jonka sisään toiminta toteutetaan. Tässä tapauksessa painiketta painettaessa muokkain lisää [row]-lyhytkoodit tekstiin. Jos muokkaimessa on sisältöä valittuna, lisätään lyhytkoodit valitun sisällön ympärille. Kuvassa 18 on JavaScript-tiedosto, jolla painike on toteutettu. Kuvassa 19 on valmis painike työkalurivissä.

```
(function() {
    tinymce.create('tinymce.plugins.trivalExtension', {
        /**
         * Funktion suorituksen yhteydessä luodaan uusi painike. Painikkeelle voidaan määritellä ikoni
         * tai käyttää otsikkoa sen tilalla. Painikkeelle määritellään myös komento joka suoritetaan kun sitä painetaan
         */
        init : function(ed, url) {
                ed.addButton('addRowButton', {
                    title : 'Add Row',
                    image: url + '/addrow.jpg',
                    cmd : 'addRowCommand'
            });

                /**
                * Komento, joka suoritetaan kun painiketta painetaan. Palauttaa maalatun teksin ympärille lisättynä lyhytkoodit
                * joilla Lyhytkoodit-luvussa tehdyt sarakejaot voidaan toteuttaa.
                */
                ed.addCommand('addRowCommand', function() {
                    var selected_text = ed.selection.getContent();
                    var return_text = '';
                    return_text = '[row]' + selected_text + '[/row]';
                    ed.execCommand('mceInsertContent', 0, return_text);
            });
        },
```

KUVA 18: TinyMCE:n painikkeiden toteutus JavaScriptillä.



KUVA 19: Painike on lisätty työkaluriviin ja sen toiminta tarkastettu.

Painikkeen toteutus jokaiselle rivi- ja sarake-vaihtoehdolle erikseen kasvattaa työkalurivin kokoa ja tekee siitä vaikeakäyttöisemmän. Käyttäjäystävällisempiä ratkaisuja voisi olla toteuttaa painikkeen sijaan alasvetolaatikko valmiista sarakevaihtoehdoista tai yksittäinen painike, josta aukeaa graafinen ponnahdusikkuna eri vaihtoehtoja varten. Muokkaimeen on myös mahdollista lisätä omia tyylisäännöksiä add_editor_style()-funktiolla. Lisäämällä muokkaimeen samoja tyylisäännöksiä kuin sivuston julkisella puolella saadaan muokattava sisältö näyttämään mahdollisimman samanlaiselta kuin valmis, julkaistu sisältö.

7 YHTEENSOPIVUUS JA ONGELMANRATKONTA

7.1 Ohjelmointivirheet

Yksi WordPressin vahvuuksista on mahdollisuus laajentaa sivustojen toiminnallisuutta kolmannen osapuolen lisäosilla. Erilaisten lisäosien käyttö saattaa kuitenkin aiheuttaa ongelmia sivuston toiminnassa koodiristiriitojen vuoksi. Tuen saaminen varsinkin ilmaisten lisäosien valmistajilta voi olla hankalaa. Tässä luvussa käsitellään yleisiä ongelmia ja miten niiden ongelmanratkontaa kannattaa lähestyä.

Teema- ja lisäosakehityksessä on suositeltavaa nimetä kaikki funktiot, muuttujat ja luokat esimerkiksi käyttämällä teeman työnimeä etuliitteenä, jotta ne eivät aiheuta ristiriitoja kolmannen osapuolen lisäosien kanssa. Suositeltavaa on myös käyttää WordPressin omaa virheraportointitilaa kehityksen aikana. Virheraportointitilan voi aktivoida WordPress-asennuksen juurikansion wp-config.php-tiedostosta, kuten kuvassa 17.

```
68 /**
69  * For developers: WordPress debugging mode.
70  *
71  * Change this to true to enable the display of notices during development.
72  * It is strongly recommended that plugin and theme developers use WP_DEBUG
73  * in their development environments.
74  *
75  * For information on other constants that can be used for debugging,
76  * visit the Codex.
77  *
78  * @link https://codex.wordpress.org/Debugging_in_WordPress
79  */
80 define('WP_DEBUG', true);
81
82 define('SCRIPT_DEBUG', true);
```

KUVA 17: wp-config.php-tiedosto. Tavallisen virheraportointitilan lisäksi käytössä on myös JavaScript-virheilmoitukset näyttävä SCRIPT_DEBUG-tila.

Teemojen ja lisäosien JavaScript- ja PHP-koodin virheet ja ristiriidat saattavat aiheuttaa ongelmia sivustojen toiminnassa. Yleinen merkki JavaScript-ongelmista on sivuston interaktiivisten ominaisuuksien tai tietyn lisäosan toimimattomuus. Ongelman aiheuttavan lisäosan voi yrittää paikantaa poistamalla lisäosat yksi kerrallaan käytöstä ja testaamalla sivustoa poistojen välissä kunnes ongelmat häviävät. Joissakin selaimissa on myös mahdollista aktivoida JavaScript-konsoli, joka näyttää koodin virheilmoitukset.

PHP-koodin virheet saattavat aiheuttaa monia eri virhetiloja. Yleisin on kuitenkin tyhjä, valkoinen sivu sivuston julkista puolta katsottaessa. Pahimmillaan ongelma saattaa myös esiintyä sivuston ohjausnäkymässä, jolloin virhe täytyy korjata muokkaamalla sivuston koodia palvelimen hallintajärjestelmässä tai esimerkiksi poistamalla virheen aiheuttava lisäosa käyttämällä FTP-yhteyttä. Koska PHP-koodi ajetaan palvelimella, tallentuvat virheilmoitukset palvelimen lokitiedostoihin. Tutkimalla lokitiedostoja voi saada selville, missä tiedostossa ja millä rivillä virheen aiheuttanut koodi on.

Kaikki virhetilat eivät ole suoraan havaittavissa. Esimerkiksi teeman kehityksen aikana esiintyi ongelma käytettäessä sivuston tietoturvaa parantavaa iThemes Security –lisäosaa. Kun lisäosan asetuksia yritettiin muokata, antoi lisäosa virheilmoituksen tuntemattomasta sisältötyypistä. Ongelman aiheuttajaksi paljastui teeman funktiotiedostoon jäänyt ylimääräinen include-koodi, jolla funktioihin liitettiin tiedosto, joka oli aikaisemmin poistettu tarpeettomana. Vika löytyi palvelimen lokitiedostoa tutkimalla. Muita näkyviä ongelmia koodivirhe ei aiheuttanut.

7.2 WooCommerce-verkkokauppalisäosa

WooCommerce on suosittu verkkokaupparatkaisu WordPress-sivustoille ilmaisuuden ja yksinkertaisen asennuksen ansiosta. Lähtökohtaisesti WooCommerce ei kuitenkaan tue kolmannen osapuolen teemoja, vaan ehdottaa sivuston ohjausnäkymässä ylläpitäjää asentamaan WooCommercen kehittäjien toteuttamia teemavaihtoehtoja. WooCommercen kauppa-, tuote- ja luokittelusivut käyttävät omia sivupohjia, jotka eivät käytä samaa luokkarakennetta kuin Bootstrap, jolloin sisällön keskitys ja responsiivisuus eivät välttämättä toimi. Yhteensopivuusongelma voidaan ratkaista kahdella tapaa: luomalla teemakansioon uusi sivupohja woocommerce.php, joka ohittaa WooCommercen omat sivupohjat, tai poistamalla funktioilla WooCommerce-sivujen sisällön luokkarakenteen muodostavat koukut ja korvaamalla ne omilla koukuilla.

Jotta WooCommerce:n oletustyylitys toimii kauppasivuilla, on tärkeää, että sivun HTML-body-avaustunnisteeseen merkitään woocommerce-luokka. Helpoiten tämä onnistuu, kun lisätään WordPressin body_class()-funktiokutsu avaustunnisteeseen, joka lisättiin header-tiedostoon. Kun yhteensopivuusongelmat on ratkaistu, voidaan funktio-tie-

dostoon lisätä funktiokutsu ilmaisemaan tuki WooCommercelle, jolloin ilmoitus mahdollisista yhteensopivuusongelmista poistuu sivuston ohjausnäkymästä. (WooCommerce n.d.)

WooCommerce:n keväällä 2017 julkaisemaan versioon 3.0 lisättiin myös sisäiset galleria-ponnahdusikkuna- ja zoomausominaisuudet tuotekuville. Nämä voidaan ottaa teeman käyttöön lisäämällä yllä luotuun woocommerce_support()-funktioon add_theme_support()-kutsut kullekin ominaisuudelle, kuten alla kuvassa 18.

```
add_action( 'after_setup_theme', 'woocommerce_support' );
function woocommerce_support() {
    add_theme_support( 'woocommerce' );
    add_theme_support( 'wc-product-gallery-zoom' );
    add_theme_support( 'wc-product-gallery-lightbox' );
    add_theme_support( 'wc-product-gallery-slider' );
}
```

KUVA 18: WooCommerce 3.0 -versiossa tulleiden ominaisuuksien käyttöönotto ja WooCommerce-tuen ilmaiseminen teeman funktio-tiedostossa.

8 POHDINTOJA

Teeman kehittämisen tarpeellisuus omien WordPress-sivustojen toteutukseen vahvistui opinnäytetyön aikana. Kolmannen osapuolen teemojen käytön ongelmina korostuivat rajoitteiset käyttötarkoitukset. Esimerkkinä tapaukset, joissa kolmannen osapuolen teeman päälle toteutettua asiakassivustoa piti laajentaa uusilla alasivuilla, joiden ulkoasu poikkesi muusta sivustosta. Sivustolla käytössä olevan teeman responsiivisuus oli toteutettu määrittelemällä sivurakenteille absoluuttiset leveydet pikseleinä tietyillä näytönleveyksillä. Koska alasivujen oli tarkoitus olla leveämpiä kuin teeman oletusleveys, jouduttiin kaikki rakenteet muokkaamaan yksitellen leveämmäksi alasivuja varten.

Lisäosa- ja teemaohjelmoinnin kannalta tärkeäksi osoittautui etuliitteiden käyttö funktioiden, koukkujen, luokkien ja muuttujien nimeämisessä. WordPress-sivustoissa kohtaavat monen eri tahon toteuttamat koodiosaset, joten nimiristiriidat kannattaa estää käyttämällä esimerkiksi teeman nimeä etuliitteenä nimeämisessä.

Teeman kehityksen suurimmaksi ongelmaksi koitui Bootstrapin käyttö teeman päävalikkorakenteen toteutuksessa. Bootstrap ei tue WordPressin käyttämää oletusvalikkorakennetta, vaan sitä varten pitää toteuttaa oma valikkorakenne käyttäen WordPressin Walker-luokkaa. Ongelmia tuotti myös visuaalisen muokkaimen laajentaminen napilla, joka syöttää käyttäjän valitsemat lyhytkoodipätkät tekstiin. Napin toteutus tapahtui JavaScript-kielellä, mitä ei muuten tarvita WordPress-kehityksessä kuin harvoin. Koska WordPress perustuu lähinnä PHP-ohjelmointikieleen, toimiva teema on mahdollista muuten toteuttaa ilman JavaScriptiä.

Kehityksen tiukan aikataulun vuoksi opinnäytetyönä toteutettua teemaa ei raportoinnin kirjoitushetkellä ehditty testata käytännössä asiakasprojektin toteutuksessa. Tarkoitus on käyttää teemaa sopivan asiakasprojektin toteutuksessa ja suunnitella projektin kokemusten perusteella, mihin suuntaan teemaa voisi vielä kehittää.

Muutamia kehitysideoita jo löydettiin. Ensimmäinen kehitysidea on sivujen muokkausnäkymään toteutettavat valikot, joissa voidaan esimerkiksi määritellä yksittäiselle sivulle oma taustakuva tai sivuvalikko ilman, että tarvitsee tehdä uutta sivupohjaa muokkauksia varten. Toisena kehitysideana on lisätä Mukauttimeen tekstikenttiä yleisesti käytettyjä

muuttujia, kuten puhelinnumeroa tai sähköpostia, varten. Näille muuttujille tehdään lyhytkoodit, joilla kyseisiä tietoja voi lisätä sivuille niin, että ne tulevat Mukauttimeen syötetyistä arvoista. Jos esimerkiksi puhelinnumero myöhemmin vaihtuu, voidaan tällöin sen kaikki esiintymiset sivustolla päivittää samalla kertaa syöttämällä uusi numero Mukauttimeen.

LÄHTEET

Built With. N.d. CMS Usage Statistics. Luettu 3.7.2017.
https://trends.builtwith.com/cms

Lyngbø, T. 2015. 10 WordPress SEO Questions That Took Me 10 Years To Answer. Search Engine Land. Luettu 24.11.2017.
https://searchengineland.com/10-wordpress-seo-questions-took-10-years-answer-214050

McCollin, R. 2015. When to Use Bootstrap for Your WordPress Theme (And When Not To). EnvatoTuts+. Luettu 4.9.2017.
https://code.tutsplus.com/tutorials/when-to-use-bootstrap-for-your-wordpress-theme-and-when-not-to--cms-23561

W3Techs. N.d. Usage of content management systems for websites. Luettu 3.7.2017.
https://w3techs.com/technologies/overview/content_management/all/

WooCommerce. N.d. Third party / custom / non-WC theme compatibility
Luettu 14.8.2017
https://docs.woocommerce.com/document/third-party-custom-theme-compatibility/

WordPress.org. N.d.a. Theme handbook - Organizing Theme Files. Luettu 6.8.2017.
https://developer.wordpress.org/themes/basics/organizing-theme-files/

WordPress.org. N.d.b. Theme handbook - Child Themes. Luettu 5.9.2017.
https://codex.wordpress.org/Child_Themes

WordPress.org. N.d.c. I18n for WordPress Developers. Luettu 12.10.2017.
https://codex.wordpress.org/I18n_for_WordPress_Developers